Notation

discrete : UPPER CASE continuous : lower case
ln = nats
y : latent varaible / ground truth (hidden, not measurable)
x : observable variable / measurement
$\hat{y}$: output of DNN
$\odot$ element-wise multiplication
Downstream task (DN) : the task you actually want to solve (classification, ...)

General stuff

Matrix & Math basics

$(cA)^{-1} = c^{-1}A^{-1}$
$\det\left(A^{-1}\right) = (\det A)^{-1}$
$\partial X^{-1} = -X^{-1}(\partial X)X^{-1}$ (mit $XX^{-1} = I$ und $\partial(I) = 0$)
$\frac{\partial}{\partial x}x^T\mathbf{B}x = x^T(\mathbf{B} + \mathbf{B}^T)$
$\|\underline{x}\|^2 = \underline{x}^T\underline{x}$
$\log_b(P \cdot Q) = \log_b P + \log_b Q$
$\log_b P^n = n \cdot \log_b P$ so $\frac{1}{2}\log(x^2) = \log(x)$

Hessian matrix $H(\underline{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_N} \\ \cdots & \cdots & \cdots \\ \frac{\partial^2 f}{\partial x_N \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_N^2} \end{bmatrix}$

$x_{1,2} = \dfrac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
$(\frac{u}{v})' = \frac{u'v - uv'}{v^2}$
$\lim_{x\to 0} x\ln(x) = 0$

Moments of vector

mean : $\underline{\mu} = \mathbb{E}(\underline{X}) = \int \underline{x}p(\underline{x})d\underline{x} \overset{\text{discrete}}{=} \sum \underline{x}_i P(\underline{x}_i)$
correlation : $\mathbf{R} = E(\underline{X}\underline{X}^\intercal) \int \underline{x}\underline{x}^\intercal p(\underline{x})d\underline{x} \overset{\text{discrete}}{=} \sum \underline{x}_i\underline{x}_i^\intercal P(\underline{x}_i)$
covariance : $\mathbf{C} = E((\underline{X} - \underline{\mu})(\underline{X} - \underline{\mu})^\intercal) \overset{\text{discrete}}{=} \mathbf{R} - \underline{\mu}\underline{\mu}^\intercal$

Jaccard / IoU (intersection-over-union)
$0 \leqslant J = \frac{|A\cap B|}{|A\cup B|} = \frac{|A\cap B|}{|A|+|B|-|A\cap B|} \leqslant 1$
Dice coefficient
$0 \leqslant D = \frac{2|A\cap B|}{|A|+|B|}$
Chain rule of derivative
$\frac{\mathrm{d}f(g(\theta))}{\mathrm{d}\theta} = \frac{\mathrm{d}f}{\mathrm{d}g}\frac{\mathrm{d}g}{\mathrm{d}\theta}$

Hyperparameters η : configuration parameters, not adapted during training. (Gradient descent not applicable)
time-invariant shifted input signal → shifted output signal $x(n - n_0) \to y(n - n_0)$ (e.g. convolution 1D)
shift-invariant $x(n_1 - i_1, n_2 - i_2) \to y(n_1 - i_1, n_2 - i_2)$ (e.g. convolution 2D)
temporal correlation : loss of meaning if data is scrambled
Inductive bias : Bias that shapes learning in a desired way (e.g. infrastructure enables things).
- dense layer → information from all neurons → decision making
- convolutional layer → learning of local feature
- recurrent layer → learning of temporal correlations

Cross-correlation (= attention) : find known segment in long measurement
Autocorrelation (= self-attention) : find self-similarities beween segments of same signal

Neural architecture search (NAS) : automated search for good fitting DNN architecture
Causal inference/reasoning is beyond ML !
Epoche : one whole pass of all training-samples through the NN. (Consists of N/B minibatches)
$\nabla\|\underline{w}\|^2 = 2\underline{w}$
Empirical distribution : $\hat{p}(x, y) = \frac{1}{N}\delta(\underline{x} - \underline{x}(n),\ \underline{y} - \underline{y}(n))$ (δ : Dirac-Kernel)

Probabilities

PMF - $P(\cdot)$
(probability mass function)
discrete

PDF - $p(\cdot)$
(probability density function)
continuouse

Uniform distribution

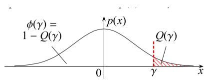
PDF : $p(x) = \begin{cases} \frac{1}{b-a} & a \leqslant x \leqslant b \\ 0 & \text{otherwise} \end{cases}$

Gaussian/normal distribution

PDF : $p(x) = \frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$
mit $\underline{x} \sim \mathcal{N}(\mu, \sigma^2)$ and $|\mathbf{C}| = \det(\mathbf{C})$ $\det(\sigma^2\mathbf{I}) = \sigma^{2d}$
Mean $E(\underline{x}) = \underline{\mu} = \frac{1}{N}\sum_{n=1}^N x_n$
Covariance $\mathbf{C} = \mathrm{E}[(\underline{x} - \underline{\mu})(\underline{x} - \underline{\mu})^T] = \frac{1}{N}\sum_{n=1}^N(\underline{x_n} - \underline{\hat{\mu}})(\underline{x_n} - \underline{\hat{\mu}})^T$
Contour lines : ellipsoids centered at $\underline{\mu}$. Direction & size of principal axes given by eigenvectors of $\mathbf{C}$

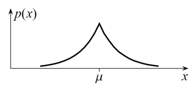
Standard Gaussian/normal :

PDF : $p(x) = \frac{1}{\sqrt{2\pi}}e^{-\frac{x^2}{2}}$
mit $\underline{x} \sim \mathcal{N}(0, 1)$
Q-func : $Q(x) = 1 - \int_{-\infty}^{x} p(z)\,dz$



Laplace distribution

PDF : $p(x) = \frac{1}{2b}\exp\left(\frac{|x-\mu|}{b}\right)$



Bernulli distribution

coin-flip $X \in \{0, 1\}$ $p = P(X = 1) > 0$
$P(x) = \begin{cases} 1 - p & x = 0 \\ p & x = 1 \end{cases}$

Cross Entropy (CE)

Entropy $H(X)$: measurement of uncertainty
Between two distributions $X \sim p$ and $Y \sim q(y)$
$H(X, Y) = H(p, q) = -\mathbb{E}_{X\sim p}\ln(q(x)) = -\int p(x)\ln(q(x))dx$
Applied to learning : $H(\hat{p}, q) = \frac{1}{N}\sum_{n=1}^N L(\underline{x}(n), \underline{y}(n), \underline{\theta})$

Kullback-Leibler-Divergence (KLD)

Between two distributions p and q
$\mathrm{D_{KL}}(p||q) = \int p(x)\ln\left(\frac{p(x)}{q(x)}\right)dx = \mathbb{E}_{\underline{x}-p(x)}\left[\ln\left(\frac{p(x)}{q(x)}\right)\right]$
with Gaussian distribution :
$\mathrm{D_{KL}}(p \sim \mathcal{N}(\mu_p, \sigma_p^2)||q \sim \mathcal{N}(\mu_q, \sigma_q^2) = \ln\left(\frac{\sigma_q}{\sigma_p}\right) + \frac{\sigma_p^2 + (\mu_p - \mu_q)^2}{2\sigma_q^2} - \frac{1}{2}$

— non-negative $\mathrm{D_{KL}}(p||q) \geqslant 0$
— equality $\mathrm{D_{KL}}(p||q) = 0$ if $p(x) = q(x)\forall x$
— asymmetry $\mathrm{D_{KL}}(p||q) \neq \mathrm{D_{KL}}(q||p)$ forward vs backward (→ NOT a distance)
— additivity $\mathrm{D_{KL}}(p||q) = \mathrm{D_{KL}}(p_1||q_1) + \mathrm{D_{KL}}(p_2||q_2)$ with $x = [x_1, x_2]^\intercal$, all x_i independent ($\to p(x) = p(x_1)p(x_2)$)

	Forward KL divergence $D_{KL}(p \parallel q)$	Backward KL divergence $D_{KL}(q \parallel p)$
	$p(x)\ln\left(\frac{p(x)}{q(x)}\right)$	$q(x)\ln\left(\frac{q(x)}{p(x)}\right)$
$p > q$	> 0	< 0
$q \to 0$	$\to \infty$	$\to 0$
$p \to 0$	$\to 0$	$\to \infty$
	Minimize $D_{KL}(p \parallel q)$ $p = 0$: doesn't care about q $p > 0$: make q close to p	Minimize $D_{KL}(q \parallel p)$ $q = 0$: doesn't care about p $q > 0$: make q close to p
	"Zero avoiding" strategy for q : $q > 0$ if $p > 0$ i.e., makes $q(x)$ broader than $p(x)$	"Zero avoiding" strategy for q : $q = 0$ if $p = 0$ i.e., makes $q(x)$ narrower than $p(x)$
Make denominator in $\ln(\cdot)$ broad to minimize D_{KL}		

$\mathrm{D_{KL}}(p||f) = -H(p) + H(p, f)$ as $H(p)$ fixed :
$\min_q \mathrm{D_{KL}}(p||q) \hat{=} \min_q H(p||q) \Rightarrow$ KLD$\hat{=}$Cross Entropy
Jensen–Shannon divergence
Symetric KLD : $\mathrm{D_{JS}} = \frac{1}{2}\left(\mathrm{D_{KL}}\left(p \parallel \frac{p+q}{2}\right) + \mathrm{D_{KL}}\left(q \parallel \frac{p+q}{2}\right)\right)$

Neuron

$y = \phi(\underline{w}^\intercal\underline{x} + b)$

Initialization

Zero initialization is bad ! (All neurons behave the same)
Must be Symmetry-breaking !
Random initialization bias : zeros ; weight : random ✓
e.g. normal or uniform distribution
He initialization
Constant activation flow (forward pass)
through $\mathrm{Var}(a_{l,i}) = M_{l-1}\sigma_{w,l}^2\sigma_{x,l-1}^2$
$\Rightarrow \sigma_{w,l} \sim \frac{1}{\sqrt{M_{l-1}}}$ M_{l-1} : Fan-in
Glorot initialization
Constant gradient flow (forward & backward pass)
$||\frac{\partial L}{\partial a}|| = \text{const}$
$\Rightarrow \sigma_l \sim \frac{1}{\sqrt{M_l}}$ M_l : Fan-out

Hyper-parameter optimization

use of a validation-set D_{val}
Grid-search : time-consuming (many params)
Bayesian optimization

Kernels

Create estimate $\hat{p}$ of $p(\underline{x})$ based on samples $\underline{x}(n)$
$\hat{p}(\underline{x}) = \frac{1}{N}\sum_{n=1}^N k(\underline{x} - \underline{x}(n))$ with kernel $k(\underline{x})$

Dirac-Kernel

$k(\underline{x}) = \delta(\underline{x}) = \begin{cases} \infty & x = 0 \\ 0 & x \neq 0 \end{cases}$ with $\int \delta(x)dx = 1$

Layers

weights : $\underline{w} \in \mathbb{R}^d$
bias : $b \in \mathbb{R}$
activation : $a = \underline{w}^\intercal \underline{x} + b \in \mathbb{R}$
activation-function : $\phi : \mathbb{R} \mapsto \mathbb{R}$
$y = \phi(a)$

> Dense Layer (ch. 4)

fully connected layer with c neurons
($c \cdot d$ weights + c bias $\Rightarrow c(d+1)$ parameters)

> Convolutional layers (ch. 7)

$A \in \mathbb{R}^{H \times W \times C_{\text{in}} \times C_{\text{out}}}$: height H, width W, channel depth C
convolution wit kernel $h(n)$: $y(n) = \sum_i h(i)x(n-i)$
2D convolutional layer 2D feature map of size $H_l \times W_l \to$ 3D tensor requires 4D kernel
Key ideas :

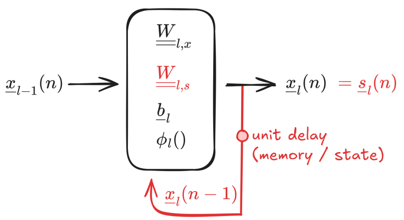
- Sparse connections : Neuron are only connected to a small subset
- Parameter sharing : Params can not be chosen independently. (In CNN : each neuron has the same set pf in- and outgoing weights.

Properties :

- Few parameters (parameter sharing), many multiplications
- sparse connections
- small receptive field (focus on local patterns)
- time-shift invariant (translation-equivariant)
- one feature per feature-map
- $H_l = H_{l-1} - k_l + 1 \;\to$ image shrinks

$a_h = \sum_{i=1}^K w_i x_{h+i-1} \quad 1 \leqslant h \leqslant H - K + 1$
Padded convolution : Zero-padding input to get same output-size as input. $p \geqslant 0$
Strided convolution : Move kernel by $S \geqslant 1$ steps (instead of 1) - f.7-9 $\Rightarrow$ Downsampling
Dilated convolution : Let kernel select only every $D \geqslant 1$'th entry (instead of 1) - f.7-10 $\Rightarrow$ Downsampling
output dimension : $\left\lfloor \frac{H_{l-1} + 2P - ((K-1)\cdot D+1)}{S} \right\rfloor + 1$
Kernel; Padding$=0$; Dilation$=1$; Stride$=1$

> Recurrent layer (ch. 8)



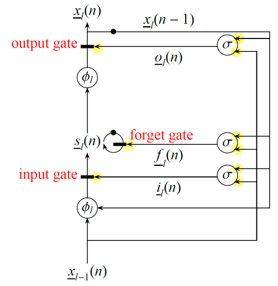
cross-neuron feedback : $\underline{\underline{W}}_{l,s}$ non-diagonal $\to$ feedback to also other neurons in this layer

LSTM layer

Gates (long short-term memory) :

- **input** (write)
- **forget** (reset)
- **output** (read)

8 weight matrices



$0 <$ signal < 1
0 : close / clear memory - 1 : open / keep content

> Attention (ch. 9)

Similarity metric

Scaled dot product : $\frac{\underline{k}^\intercal \underline{q}}{\sqrt{d}}$ (with dimension d)

cosine similarity : $0 \leqslant \frac{\underline{k}^\intercal \underline{q}}{||\underline{k}|| \cdot ||\underline{q}||} \leqslant 1$

Single-query attention

query $q \in \mathbb{R}^d$ (d interval dimension)
$N \times$ key-value pairs $k_i, v_i \in \mathbb{R}^d$, $\underline{\underline{K}} = [k_1, .., k_N]$, $\underline{\underline{V}}$
calc similarities $a_i = s(\underline{q}, \underline{k}) \to$ importance weights $\alpha_i = \text{softmax}(a_i)$
$\to$ refined v_i for $\underline{q}$: attention$(\underline{q}, \underline{\underline{K}}, \underline{\underline{V}}) = \sum_{i=0}^N \alpha_i v_i$

Multi-query attention

multiple queries $\underline{\underline{Q}} \in \mathbb{R}^{d \times M}$
$\to M$ attention vectors

Multi-head attention

h heads (each with own dot-product attentions - with own weights from) $\underline{\underline{W}}_i^Q, \underline{\underline{W}}_i^K, \underline{\underline{W}}_i^V$

Skip connections (Shortcut)

over one or multiple layers (identity paths) backward pass :
$\underline{x}_l = \phi_l(W_l x_{l-1} + b_l) + x_{l-1}$
$\frac{\partial \underline{x}_l}{\partial \underline{x}_{l-1}} = \frac{\partial \phi_l()}{\partial \underline{x}_{l-1}} + I$
Residual connections mandatory for deep networks!

Normalization

(weight normalization ($\underline{\underline{W}}$)) and activation normalization ($\underline{\underline{A}}$)

- instance normalization (across single channel)
- batch normalization (across same channel whole batch)
- layer normalization (across channels)

Downsampling layer

Strided convolution : *(see "Convolutional layers")*
Pooling : max pooling / mean pooling / l_2-norm pooling
$\to$ translation invariance!

Upsampling layer

interpolation, unpooling, deconvolution

Reshaping layer

flatten layer $\underline{\underline{X}} \in \mathbb{R}^{H \times W \times C} \mapsto \text{vec}(\underline{\underline{X}}) \in \mathbb{R}^{HWC}$
global average pooling (GAP) layer reduce each channel/feature to $\mathbb{R}^1$

Activation functions

linear / identity :
used in output-layer of regression.

Unit step
$\phi(a) = \begin{cases} 0 & a < 0 \\ 1 & a > 0 \end{cases}$ not differentiable!

Sign function
$\phi(a) = \begin{cases} -1 & a < 0 \\ 1 & a > 0 \end{cases} = 2u(a) - 1$ not differentiable!

Sigmoid
$\phi(a) = \sigma(a) = \frac{1}{1+e^{-a}}$
$\frac{\mathrm{d}}{\mathrm{d}a}\phi(a) = \phi(a)(1 - \phi(a))$

- $0 \leqslant \phi(a) \leqslant 1 \;\to$ good for probabilities
- symmetry
- easy computable derivative

Hyperbolic tangent
$\phi(a) = \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$
$\frac{\mathrm{d}}{\mathrm{d}a}\phi(a) = 1 - \phi(a)^2$

- like sigmoid with other output range

ReLU Rectified linear unit
$\phi(a) = \text{Relu}(a) = \begin{cases} a & a > 0 \\ 0 & a \leqslant 0 \end{cases} = \max(a, 0)$

$\frac{\mathrm{d}}{\mathrm{d}a}\phi(a) = \begin{cases} 1 & a > 0 \\ 0 & a < 0 \\ 1 \text{ or } 1 & a = 0 \end{cases}$

- very simple
- easy derivative
- zero-gradient for $a < 0$ (bad) $\Rightarrow$

  Leaky ReLU $= \begin{cases} a & a > 0 \\ 0.01a & a \leqslant 0 \end{cases}$

Softmax
$\phi(a) : \mathbb{R}^c \mapsto \mathbb{R}^c$
$\phi(a) = [\phi_1(a), \ldots, \phi_c(a)]^\intercal$ with $\phi_i(a) = \frac{e^{a_i}}{\sum_{j=1}^c e^{a_j}}$

$\frac{\mathrm{d}}{\mathrm{d}a_j}\phi_i(a) = \begin{cases} -\phi_i(a)(1 - \phi_i(a)) & i = j \\ -\phi_i(a)\phi_j(a) & i \neq j \end{cases} = \phi_i(a)(\delta_{ij} - \phi_j(a))$

Binary case : $\phi_1(a) = \sigma(a_1 - a_2)$ and $\phi_2(a) = 1 - \phi_1(a)$
Used for output-layer (normalization of classification-problems)

Loss functions
Loss $l(\underline{x}, \underline{y}, \underline{\theta})$: quality of prediction of one pair
Cost function $L(\underline{\theta})$: average loss over all training samples
$\underline{y} = f(\underline{x}; \underline{\theta}) + \text{noise}$ $\quad \underline{y}|\underline{x}$ Distribution of y given x

L_2-loss
mean square error (MSE) - (white Gaussian noise $\mathcal{N}(0, \sigma^2 I)$)
$l(\underline{x}, \underline{y}; \underline{\theta}) = ||\underline{y} - f(\underline{x}; \underline{\theta})||^2$
$L(\underline{\theta}) = \frac{1}{N} \sum_{n=1}^{N} ||\underline{y}(n) - f(\underline{x}(n); \underline{\theta})||^2$

L_1-loss
mean absolute error (MAE) - (Laplace distribution)
$l(\underline{x}, \underline{y}; \underline{\theta}) = \frac{1}{b} ||\underline{y} - f(\underline{x}; \underline{\theta})||_1 + \text{const}$
$L(\underline{\theta}) = \frac{1}{N} \sum_{n=1}^{N} ||\underline{y}(n) - f(\underline{x}(n); \underline{\theta})||_1$

Categorical CE loss
$y \in \{\underline{e}_1, \ldots, \underline{e}_c\}$ (class-labels one-hot coded) **for classification**
p is approximated by $Q(\underline{y}|\underline{y}; \underline{\theta}) = \prod_{i=0}^{c} f_i^{y_i}$
$l(\underline{x}, \underline{y}; \underline{\theta}) = -\ln Q(\underline{y}|\underline{y}; \underline{\theta}) = -y^\intercal \ln(f(\underline{x}; \underline{\theta}))$ (improvement : focal loss
$l(\underline{x}, \underline{y}; \underline{\theta}) = -y^\intercal (1-f)^\gamma \odot \ln(f(\underline{x}; \underline{\theta})))$

↑ distribution-based loss
↓ region-based loss

IoU and Dice loss
IoU : $1 - J = 1 - \frac{|A \cap B|}{|A|+|B|-|A \cap B|}$
Dice : $1 - D = 1 - \frac{2|A \cap B|}{|A|+|B|}$
Independent of region of background.
For multi-class segmentation : $J = \frac{1}{c} \sum_{i=1}^{c} J_i$
$|A \cap B| = \sum_h \sum_w \underline{y}_{hw} \odot \hat{\underline{y}}_{hw} =: \alpha$
$|A| + |B| = \sum_h \sum_w \underline{y}_{hw} + \hat{\underline{y}}_{hw} =: \beta$
Soft IoU and Dice loss : $J = \frac{\alpha + \epsilon}{\beta - \alpha + \epsilon} \Rightarrow$ no $\div 0$
Use Soft for training!

Important networks
MNISTnet3 : CNN for digit recognition on MNIST
CIFAR10net : CNN for image recognition (on CIFAR-10)
ResNet : stack of residual blocks
U-Net : semantic image segmentation : symmetric encoder-decoder & residual connections
Vision transformer (ViT) : transformer for image classification patch tokenization (with embedding); "transformer encoder"; "MLP head" (for classification)
SimCLR : contrastive learning with linear classification head
CLIP Contrastive language image pretraining **:**
a) → few-shot transfer : (few labeled images) backbone for image representation
or b) → zero-shot transfer : largest similarity between text and image
Pix2Pix : cGAN; translates between paired images

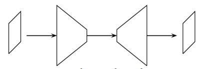
Visualization techniques
Grad-CAM : Gradient-weighted class activation mapping
→ important regions

DNN architectures
feedforward multilayer NN
or **Multilayer perceptron (MLP)**

Convolutional Neural Networks (CNN)
for image and time-series
lower memory complexity + hirarchical feature learning + time-invariant
$\underline{\underline{X}}_{l-1} \rightarrow \left| \begin{array}{c} \underline{\underline{W}}_l \\ \hline \underline{b}_l \\ \hline \underline{\underline{\Phi}}_l \end{array} \right| \rightarrow \underline{\underline{X}}_l$
$\underline{\underline{W}}_l$ Kernel tensor, $\underline{b}_l$ bias vector
Number of parameters : $\approx K_l^2 C_{l-1}$ → quiet large

encoder-head
image → number (classification)

encoder-decoder
image → image (segmentation)



Recurrent Neural Networks (RNN)
NN with feedback - nonlinear extension of IIR filters. (recursive)
→ memory of neurons : **sequential input-data** (temporal correlation)
Training : SGD with $\underline{\theta}^{t+1} = \underline{\theta}^t - \gamma^t \nabla L(t, \underline{\theta})|_{\theta=\theta^t}$ (Difficult to train, vanishing gradients, hard to parallelize → sequential calculations)
$\underline{s}_l(n) = \underline{\phi}_l \left(W_{l,x} \underline{x}_{l-1}(n) + W_{l,s} \underline{s}_l(n-1) + \underline{b}_l \right)$

Bidirectional recurrent neural networks (BRNN)
minibatch in forward- and backward-time direction in two separate recurrent layers; concatenate outputs (→ next layer has past + future info)

Long short-term memory (LSTM)
Most successful type of RNN
Replace recurrent neuron with **LSTM-cell(/neuron)**
$\underline{s}_l(n) =$
$\underline{f}_l(n) \odot \underline{s}_l(n-1) + \underline{i}_l(n) \odot \underline{\phi}_l \left(W_{l,sx} \underline{x}_{l-1}(n) + W_{l,so} \underline{x}_l(n-1) + \underline{b}_l \right)$
→ no vanishing gradient in long memory
Peephole-LSTM : gates depend on prev. state $\underline{s}_l(n-1)$
more complex (11 weight matrices)
Gated recurrent unit (GRU) : lower complexity
simplified : 2 gates (reset, update)

Transformer
multi-head self-attention; general purpose
- parallel computing (no recurrence)
- short- and long-range dependencies
- very high computation complexity (quadratic $\mathcal{N}^2$)
- very high memory complexity
- requires more training data

Tokenization : divide text into known units (tokens) - by NN (e.g. *Word2Vec*)
Positional encoding required for consideration of sequential order
Feedforward dense layer : relationships within sequence
For images : tokens for patches based on flattened 2D image-slices

Self-supervised learning
To solve data-inefficiency of NN
Self-supervised learning is representation-oriented **Self-supervised representation learning (SSL)** unlabeled dataset; mostly for pretraining
Semi-supervised learning with small dataset; reduce labeling effort
Split training in **pretraining** : model trained on other (more general?) dataset. Better than random params!
Later **finetuning** with small labeled dataset (for downstream task) of whole model, or only head (with others params frozen)
Foundation models (FM) are general-purpose models, trained on task-agnostic data

Autoencoder (AE)
encoder-decoder architecture; simplest SSL; undercomplete (inner dimension $c \ll d$ input/output)
input → encoder → latent variable z (hidden, compressed representation) → decoder → loss against input $\underline{y} = \underline{x}$

Pretext task (PT)
Artificially generated supervised task
self-reconstruction (autoencoder) $\in$ pretext task
example-tasks : image rotation, colorization, masking, super-resolution.

Contrastive learning
learn contrast between similar/dissimilar samples
samples are generated : crop, resize, rotate, flip, color distortion, cutout, gaussian noise/blur, other filters, ...
→ No task, no $\underline{y}$, no supervised loss (loss in representation space)
Goal for representation space $\underline{z}$:
similar/dissimilar samples (positives/negatives) pulled/pushes to/from each other

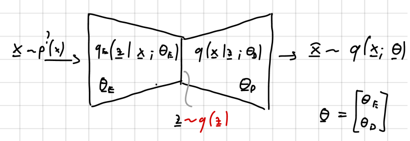
Transfer learning
embedded systems : one model for one task (due to system-restraints)
With **source task** $\mathcal{T}_s = \{X_s, Y_s, p_s(\underline{x}, \underline{y})\}$
Transfer learning applicable to **Related tasks** ($X_s = X_t, Y_s \neq Y_t$) and **Distribution/Domain shift**
($X_s = X_t, Y_s = Y_t, p_s(\underline{x}, \underline{y}) \neq p_t(\underline{x}, \underline{y})$ - like cross sensor, cross daytime, ...)
New task ($X_s \neq X_t$) is too difficult!
Related tasks : Freeze backbone and train new head - or - multitask-training
Distribution/Domain shift :
- Domain adaptation (Transform new task to source-domain)
- Continual learning (model must perform well in all new tasks), no catastrophical forgetting!

Generative models

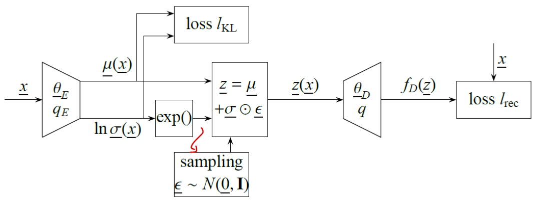
learn joint distribution $p(\underline{x}, \underline{y})$ (instead of focusing on decision boundaries like discriminative models) to generate new data according to $p(\underline{x}, \underline{y})$

Variational autoencoder (VAE)

= AE with requirement : z need to be known distribution
→ after training : draw sample from z, use decoder to generate output



$q(\underline{x}, \underline{\theta}) = \int q(\underline{x}, \underline{z}; \theta) d\underline{z} = \int q(\underline{x} \mid \underline{z}; \theta) \cdot q(\underline{z}) d\underline{z}$ for solution : replace $-\ln q(\underline{x}; \underline{\theta})$ by its variational upper bound ; use stochastic encoding + reparametrization (to enable backpropagation of gradient through sampling unit)



Generative adversarial network (GAN)

More powerful than VAE ;
generator (G) vs discriminator (D), while generator has packpropagation from discriminator.
min-max optimization : $\min_{\underline{\theta}_G} \max_{\underline{\theta}_D} L(\underline{\theta}_G, \underline{\theta}_D)$
with $L(\underline{\theta}_G, \underline{\theta}_D) =$
$\mathbb{E}_{\underline{x} \sim p_{\text{data}}} \ln\big(D(\underline{x}; \underline{\theta}_D)\big) + \mathbb{E}_{\underline{z} \sim p_{\text{noise}}} \ln\big(1 - D(G(\underline{z}; \underline{\theta}_G); \underline{\theta}_D)\big)$
Ping-Pong training :
- one minibatch updating D ($\max_{\underline{\theta}_D} L(\ldots)$) → stoch grad ascent
- one minibatch updating G ($\min_{\underline{\theta}_G} L(\ldots)$) → stoch grad descent

→ Nash-equilibrium (saddle point)
difficult ; mode collapse possible
Generate realistic (fake) samples : data augmentation for training ; data generation without privacy concerns.
No control over output (from noise $\underline{z}$) - no labels used
→ Extension : **conditional GAN (cGAN) :**
+ class label input $\underline{y} \sim p_{\text{label}}$
D accepts only if real AND matches label
→ Extension : **Paired image translation :**
On paired data $(\underline{x}, \underline{y})$ (e.g. semantic segmentation, day-night)

Diffusion model

high quality ; simpler training than GAN ; high complexity ; long training **Forward process (diffusion process) :**
Input $\underline{x}_0$ to Gaussian noise $\mathcal{N}(0, I)$ by sequence of first-order Markov processes
Reverse process (generation process) :
random sample $\underline{x}_T \sim \mathcal{N}(0, I)$ backwards through Markov process

Gradient Descent

local search ! $-\underline{\nabla} f(\underline{x})$ orthogonal to contour line (towards descent)
γ : step-size / learning rate
$\underline{\theta}^{t+1} = \underline{\theta}^t - \gamma^t \nabla L(\underline{\theta})|_{\underline{\theta} = \underline{\theta}^t}$
No need to calc Hessian → much simpler

Stochastic Gradient Descent (SGD)

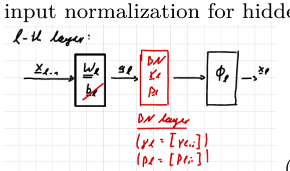
Batch gradient descent - more noisy than $L(\underline{\theta})$
Use Minibatch of size $B << D_{\text{train}}$
$\underline{\theta}^{t+1} = \underline{\theta}^t - \gamma^t \nabla L(t; \underline{\theta})|_{\underline{\theta} = \underline{\theta}^t}$
↑ minibatch size ⇒ ↑ less noise, convergence, parallel processing

... with Momentum

improvement to SGD : $\underline{\theta}^{t+1} = \underline{\theta}^t + \Delta \underline{\theta}^t$
$\underline{\theta}^{t+1} = \underline{\theta}^t - \gamma^t \nabla L(t; \underline{\theta})|_{\underline{\theta} = \underline{\theta}^t} + \beta \Delta \underline{\theta}^{t-1}$
mit $\Delta \underline{\theta}^t = \beta \Delta \underline{\theta}^{t-1} - \gamma^t \nabla L(t; \underline{\theta})|_{\underline{\theta} = \underline{\theta}^t}$ regular SGD
$0 \leqslant \beta \leqslant 1$ $\beta = 0 \to$ SGD
reduce noise in stochastic gradient ; reduce oscillation ; **accelerate** ill-conditioned **convergence Nesterov Momentum** + look-ahead gradient (ch. 5.5)

Input normalization

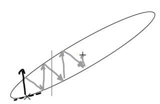
covariate shift : change of distribution of input data over time. or large offset between channels.
Input normalization over whole dataset (individually per channel)
$x_i = \frac{x_i - \mu_i}{\sigma_i}$
zero-mean unit-variance normalization decouples layers → easier training
Batch normalization : fix internal covariance shift over layers and time
input normalization for hidden layers and each minibatch



(During inference exponentially weighted averages of the training values are used)

Optimization Difficulties

- stochastic gradient
- ill conditioning
contour lines curvatures strongly different

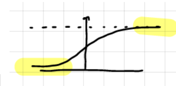


- covariate shift
- saddle point / plateau
verry slow convergence, as $\nabla L(\underline{\theta}) \approx 0$ - some parameters have a very small influence
- sensitive to step size
slow convergence vs. oscillation (with overshoots) and no convergence
⇒ learning rate decay :

1. Static schedule independent of $\underline{\theta}^t$: (Step decay, Inverse time decay, Exponential decay)
2. Adaptive schedules :
   RMSprop (root mean square propagation)
   Adam (adaptive moment estimation)
   AdaGrad (adaptive gradient algorithm)

- local minimum
- vanishing gradient (or exploding)
Backpropagation of "error vector" $J_L(\underline{a}_L) = \frac{\partial L}{\partial \underline{a}_L}$

If $\forall L : ||J_{a_{l+1}}(a_l)|| \gtrless 1 \quad \to_{l \to 1} ||J_L(a_l)|| = \begin{matrix} \infty \\ 0 \end{matrix}$
More serious for deep layers.



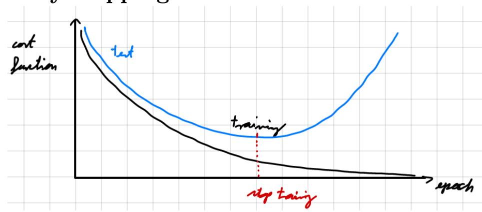
ReLU less serious that e.g. Sigmoid
other options :
- gradient clipping
- better optimization algorithm (adam)
- skip-connections in architecture

Regularization

techniques to prevent overfitting

Model capacity

Ability of themodel to learn amapping
too simple model → underfitting
too complex model → overfitting
weight norm penalty regularized cost function :
$L_r(\theta) = L(\theta) + \sum_l \lambda_l P(W_l)$ with penalty-term P and regularization parameter λ
(large weights tends to overfitting)
early stopping



data augmentation use artificial but realistic training samples to increase training-material → better model
modify data without changing class labels :
translation/rotation/scaling/flip, added noise, modified colors / textures, use of image patches, ...
ensemble learning train different models (training data subsets, architecture, cost functions, optimizer, ...)
combine models (regression : average ; classification : voting)
dropout (Implicit ensemble learning method)
randomly removes neurons based on dropout rate d_l. outgoing weights corrected by $(1 - d_l)$

Model reduction

Reduce computational/memory complexity and power consumption
Low-rank factorization : $\underline{\underline{W}} \in \mathbb{R}^{M \times N} \approx \underline{\underline{AB}} \in \mathbb{R}^{M \times K} \cdot \mathbb{R}^{K \times N}$
reduces MN to $(M+N)K$ multiplications
Pruning : Force ≈ zero columns/rows in $\underline{\underline{W}}$, remove the corresponding neurons
Quantization : reduce word length